JOHNS HOPKINS
APPLIED PHYSICS LABORATORY

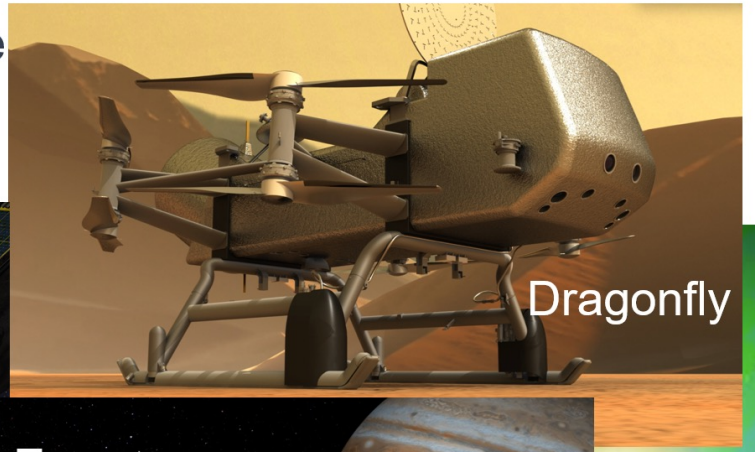
A Novel Encoder for DC-Balanced SpaceWire

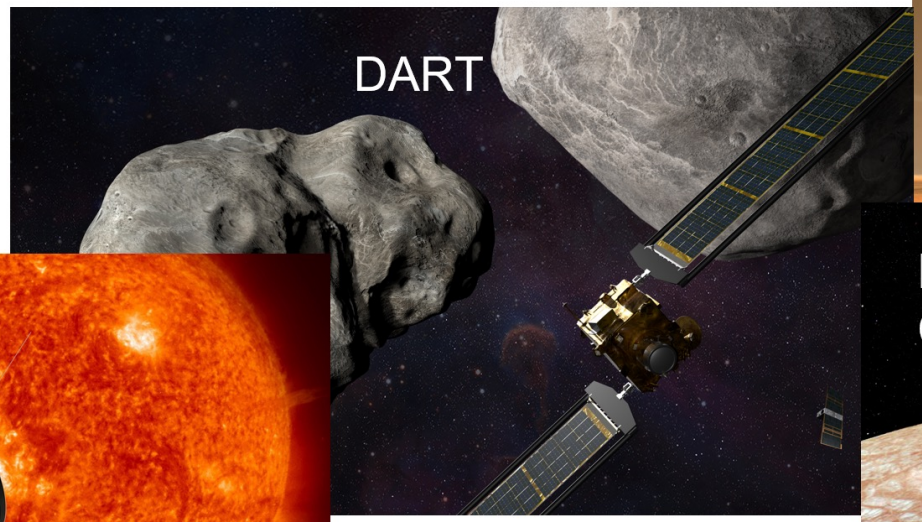
Christopher M. Rose, Steve Cho, Matthew Gile, Kirk Volland, Jarrett Wehle

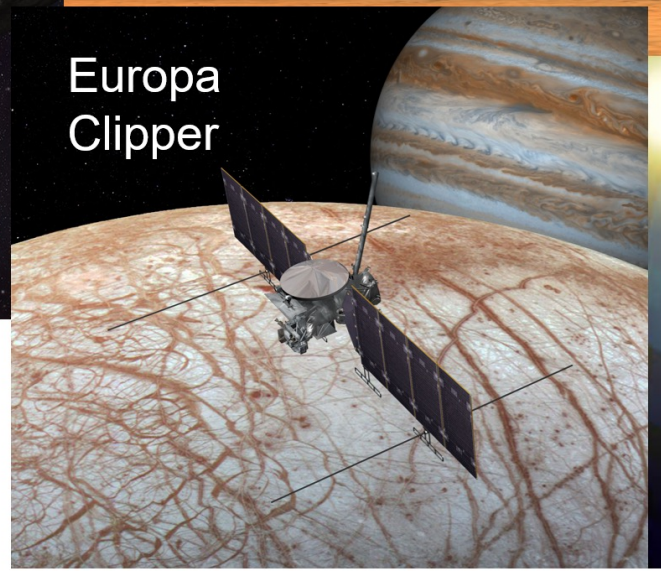
Christopher M. Rose
Senior Professional Staff II
christopher.Rose@jhuapl.edu

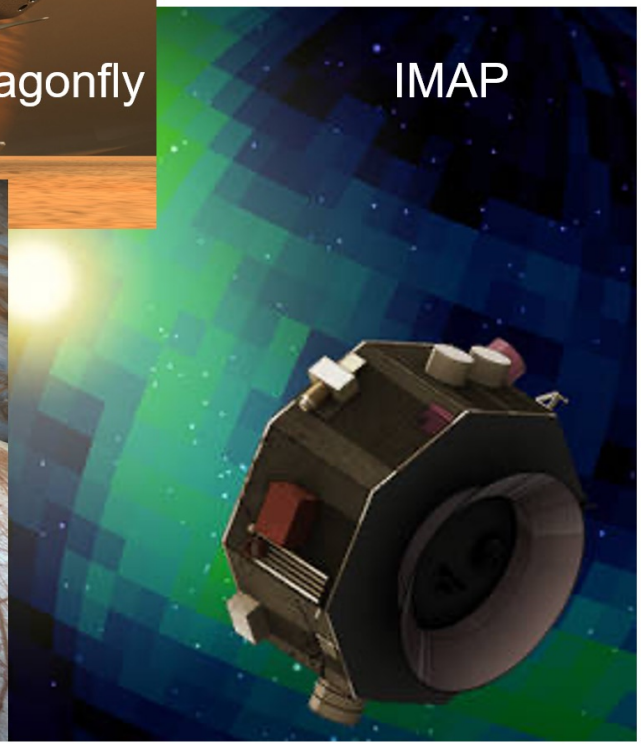
Introduction

- Chris Rose — avionics, Space Exploration Sector of JHUAPL
- first internal research into SpaceWire in 2009
- SpaceWire IP by: GSFC, Cobham, homemade
- Used in several NASA missions:



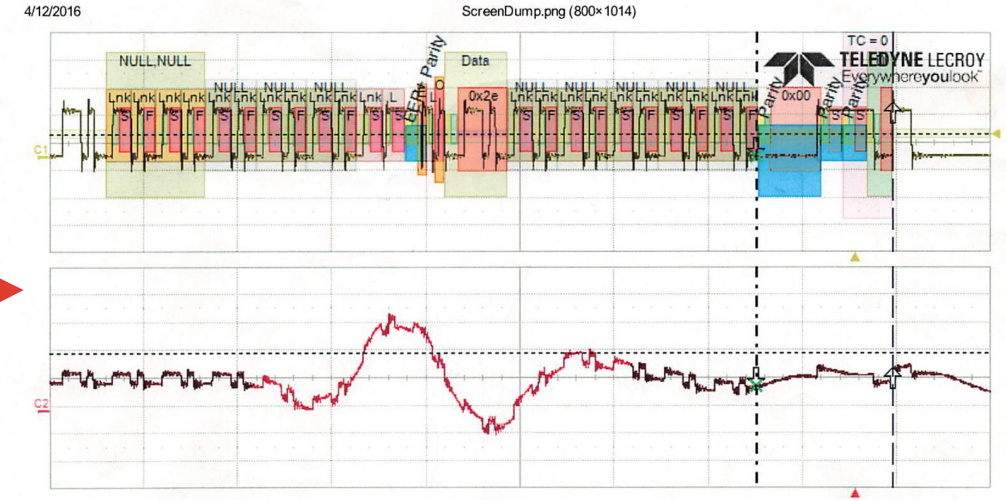






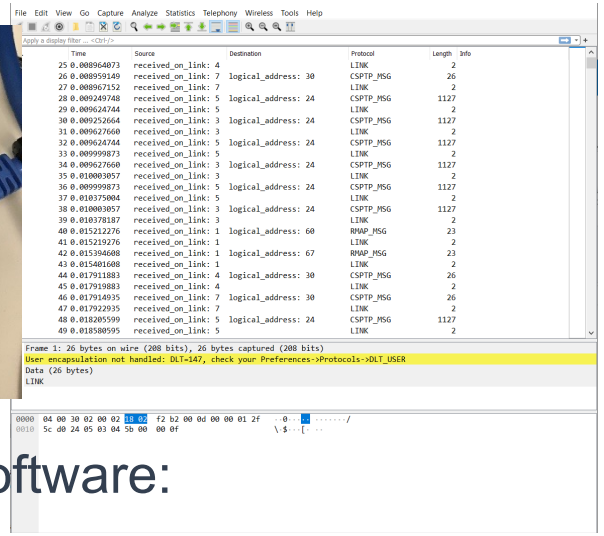
DC-Balancing to enable AC-Coupled SpaceWire

- DC-balanced signal: count the # of 1's and of 0's in a signal — the difference between # of 1's and of 0's is called the disparity
  - must not ever exceed some 'maximum disparity'
  - SpaceWire's DATA and STROBE signals each need to be balanced individually
- AC-coupled: bulk of the signal does not determine the received data;
  - Instead, it's the change in the signal that is significant.
- DC-balanced signaling is an enabler for AC-coupling
- AC-Coupling can solve several design problems:
  - Mismatched GND values across a SpaceWire link
  - Plasma charging in spacecraft exterior and interior
  - Large common mode offsets between GSE and FLT
  - Remote components or instruments, i.e. on booms
  - LVDS error propagation *(Larsen 2010, "Theory and Operation of Fault Propagation…")*



Motivation

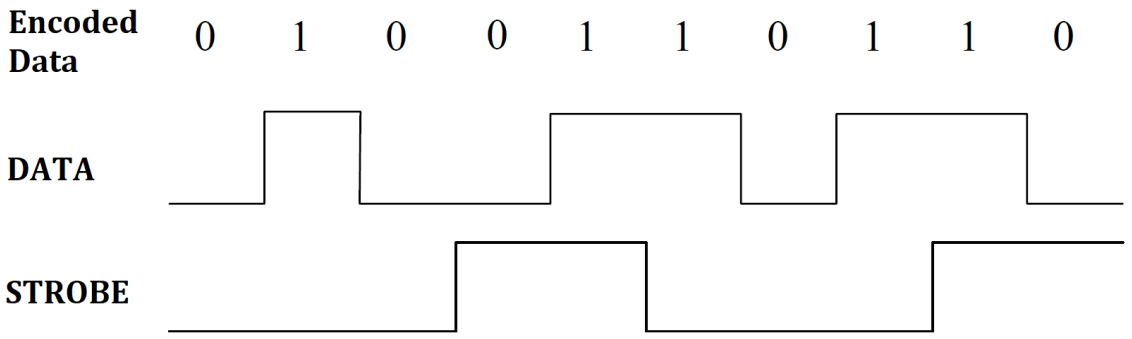
- Block encoders are commonly used for DC-balancing – D/S encoding complicates this.
- *Kissin and Rakow, 2016* offers several other approaches.
- In all of these, either the protocol is deliberately broken or the contents of encoded packets are illegible — engineers can't read the packets 'on the line'.



- APL has large existing infrastructure of SpaceWire equipment and software:
  - Link analyzers, data recorders, protocol checkers, 'bricks', WireShark, testbeds…
- APL also has a not-insignificant amount of experience with SpaceWire.
- How could we create an encoder that is DC-balanced and whose output still legible to the existing infrastructure?

SpaceWire at the signaling level

- SpaceWire = only DATA and STROBE signals, “Data-Strobe Encoding”
- No separate clock signal; clock is recovered by (DATA XOR STROBE) at the receiver
- Every transition of DATA or STROBE implies an active clock edge (DDR)
- NOTE that if no transition happens, no new data is decoded

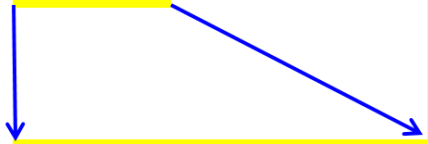


Data-Strobe Encoding: every transition is a new bit

- Idea: use this feature to balance SpaceWire signals
- Change the timing of DATA and STROBE, not their values

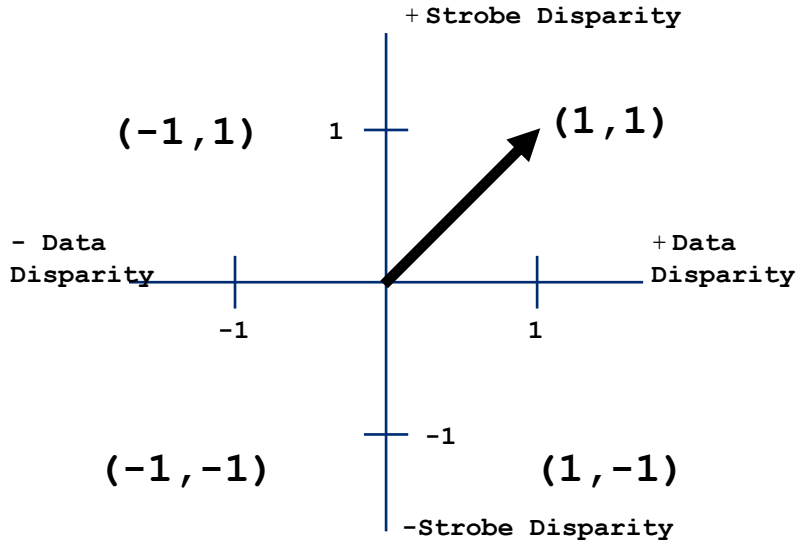
Simple Example: Cycle-Stretching a SpaceWire NULL

											Disparities
DATA	1	1	1	1	0	1	0	0			+2
STROBE	1	0	1	0	0	0	0	1			-2
DATA'	1	1	1	1	0	1	0	0	0	0	0
STROBE'	1	0	1	0	0	0	0	1	1	1	0



- A NULL character is a sequence of eight bits of DATA and STROBE
- Disparities: it has an excess of 1's in DATA, and too many 0's in STROBE
- We will find a cycle with the opposite values, and extend it, 'stretch it', until the disparities are zero.
- Now we have a DC-balanced NULL, ready for AC-coupling
- The receiving SpaceWire node just sees a NULL with some added delay
- No extra decode required(!)

One more step—Split Difference



- Think of each DATA,STROBE pair as creating a vector on a plane that has data disparity as one axis, and strobe disparity as the other axis.
- Each cycle of SpaceWire signaling moves the cumulative disparities in both these dimensions
- We can't make unit vectors by cycle stretching; we can only make diagonal vectors.
- When we cycle-stretch, our goal is to move the cumulative disparities back towards (0,0) on this plane.

(DATA, STROBE)	(Data Disparity, Strobe Disparity)
(0,0)	(-1,-1)
(0,1)	(-1,1)
(1,0)	(1,-1)
(1,1)	(1,1)

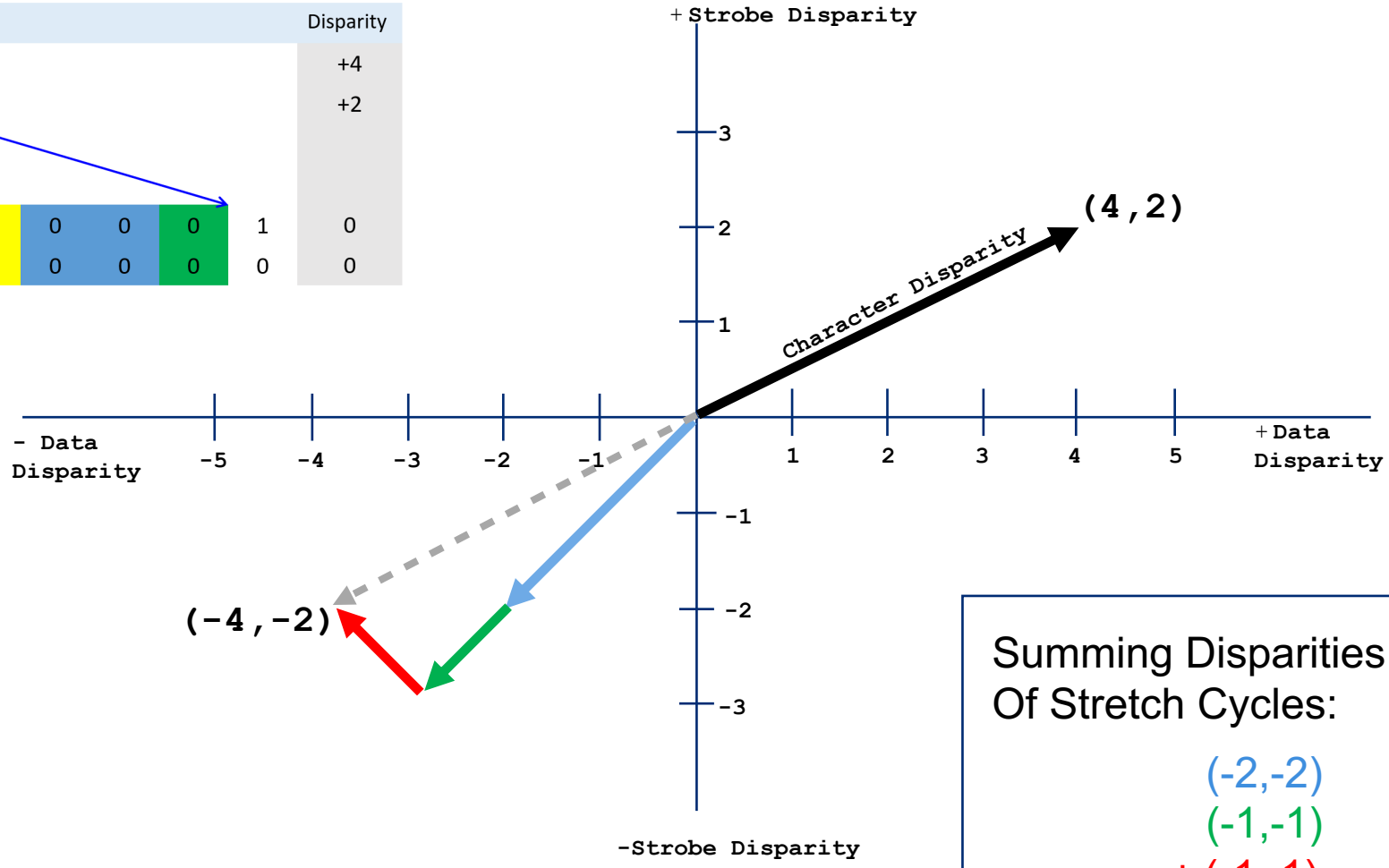
Cycle-Stretching with Split Difference Method (example)

															Disparities
DATA	1	0	1	1	1	1	1	0	0	1					+4
STROBE	1	1	1	0	1	0	1	1	0	0					+2
DATA'	1	0	0	1	1	1	1	1	0	0	0	0	0	1	0
STROBE'	1	1	1	1	0	1	0	1	1	0	0	0	0	0	0

The sum of the disparities of the stretch cycles equals the 180 degree rotation of the original character's disparity.

Cycle-Stretching with Split Difference (example, cont'd)

															Disparity
DATA	1	0	1	1	1	1	1	0	0	1					+4
STROBE	1	1	1	0	1	0	1	1	0	0					+2
DATA'	1	0	0	1	1	1	1	1	0	0	0	0	0	1	0
STROBE'	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0



The sum of the disparities of the stretch cycles equals the 180 degree rotation of the original character's disparity.

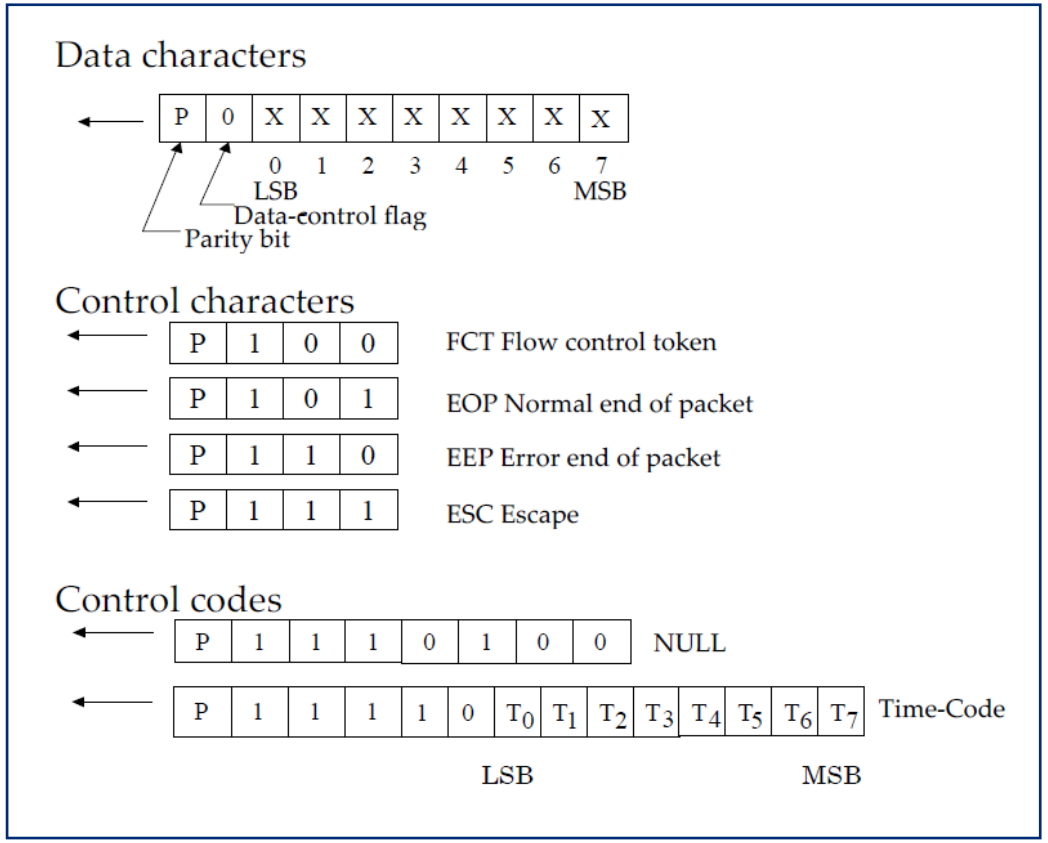
Summing Disparities Of Stretch Cycles:

(-2,-2)
(-1,-1)
+ (-1, 1)

(-4,-2)

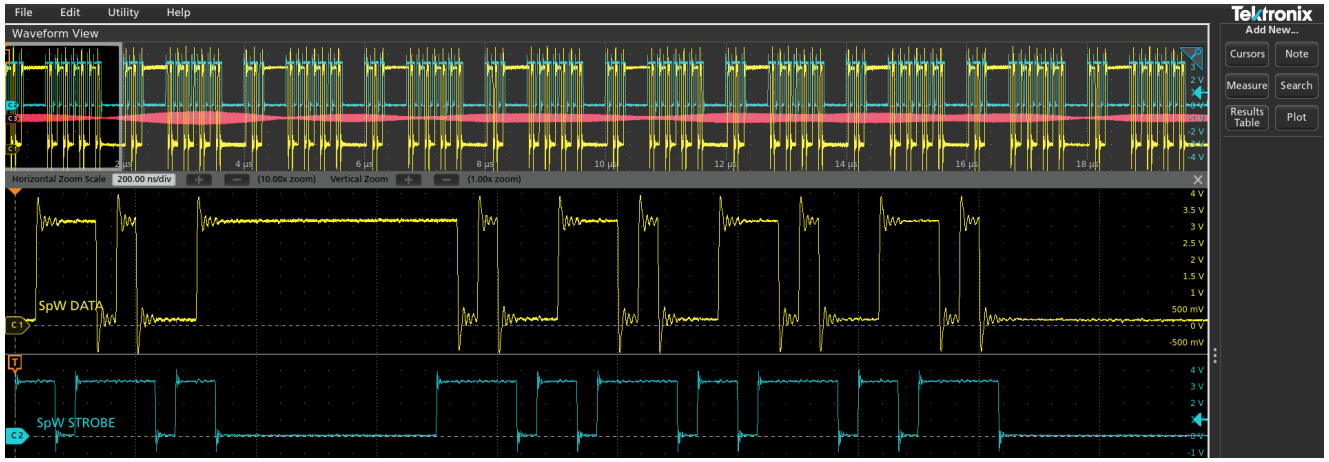
Cycle-Stretching with the Split Difference Method—Considerations:

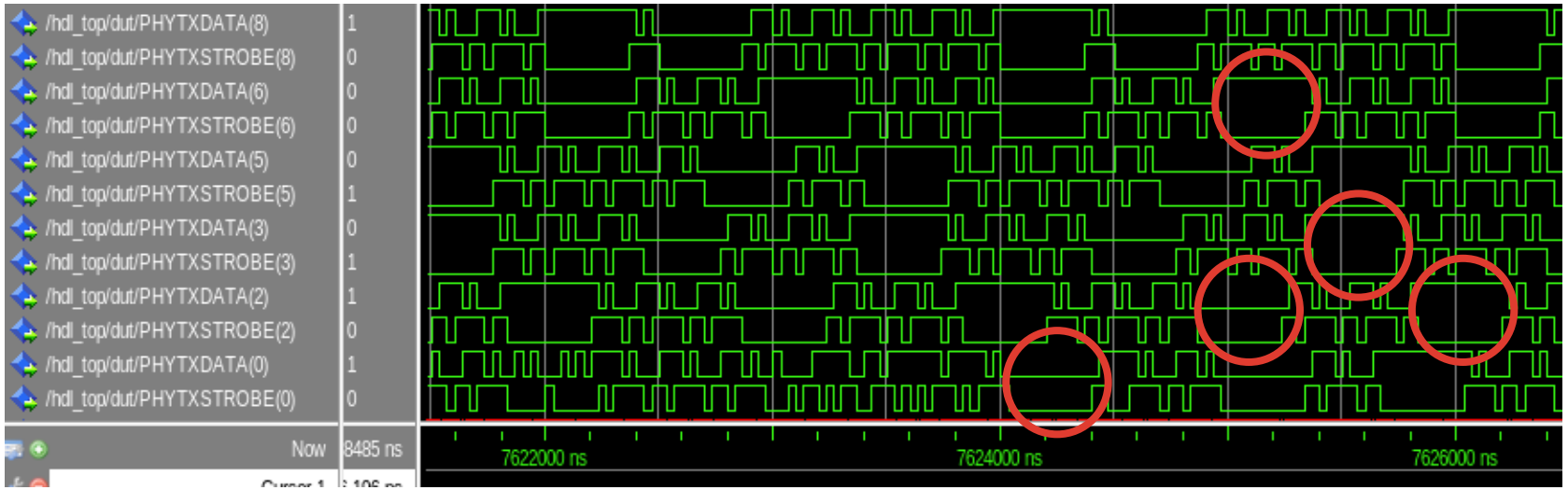
- Some characters don't have all combinations of DATA and STROBE values available for stretching.
- Some differences in disparities are not equally divisible by two.
- Some characters—particularly time codes—should not be cycle-stretched at all.
- In all of these cases, we carry over any remaining disparity into the next character.
- Some maximum # of stretch-cycles in series needs to be enforced.



Encoder demonstration

- We implemented a 10-Port IMAP router with added cycle-stretching feature (ProASIC3E)
  - Per-character cycle-stretch encoder requires little additional hardware
  - Configuration registers were added to enable/disable individual links, and to adjust several parameters
- Running automated board tests with no errors (IMAP avionics, May 2021)





UVM test suite runs router verification with thousands of packets against cycle-stretching router, with no errors

Efficiency

$$\text{Efficiency} = \frac{\text{Total number of bits before encoding}}{\text{Total number of bits after encoding}}$$

- Efficiency of cycle-stretch methods are bounded by Manchester encoding, e.g. 1/2, or 0.50
- Efficiency of simple cycle-stretch encoder is dependent on the statistical qualities of the input data. (Block encoders do not, Instead, they have a worst case baked in.)
- Packets with DATA that is all 0's, or of all 1's, are near-worst cases
- For some tests using randomized input data, we measured efficiency of ~0.66
- For comparison, block encoders have clearly defined efficiencies
- An 8b/10b block encoder has an efficiency of $\frac{8}{10} = 0.80$
- A 10b/12b block encoder has an efficiency of $\frac{10}{12} = 0.8\overline{3}$
- Those are pretty good!

How can we improve the efficiency of cycle-stretch encoders?

- We made a C model of a cycle-stretch encoder.
- We allowed the encoder to have access to the next N characters ('look-ahead') to be sent.
- Developed two 'families' of look-ahead, cycle-stretching encoders.

1.) a Naïve Lookahead Encoder:

- Take the difference between disparity D_0 and D_N

  (This is the disparity of the current character, and the cumulative disparity calculated for N characters in the future)

- Divide that difference by N, and round down.

  (Save the residue and pass it forward to the next character's calculations)

- Zero out that disparity by cycle-stretching the current character.
- Do this for both DATA and STROBE disparities simultaneously using split difference method.

Could we improve…? (cont'd)

2.) Rolling Average Lookahead Encoder:

- Sum the disparities for the first N characters to make rolling averages for D_D and D_S
- Add each new character's disparities to these rolling averages
- Subtract the disparities for the outgoing Tx character from the rolling averages
- For each character, divide rolling averages by N
- Zero out these disparities by cycle-stretching the current (outgoing) character.

Lookahead Encoders: Initial Findings

- More logic required to implement
  - but assumed to be using existing TX FIFO
- Can’t always see N characters into the future
- Efficiencies definitely vary with respect to N (bigger N is better)
- Look-ahead encoders have higher efficiencies, **BUT**
- Running a check on the outputs, we found disparities accumulating over time
- Why? Averaging methods are 'leaky'
  - There are always rounding errors
  - There are initialization and termination issues to resolve
  - These algorithms are not perfect

Efficiency Improvement: A Second Stage

IDEA: The logic that is checking the disparities of the output stream *could be feed a 'second stage': a per-character cycle-stretcher*

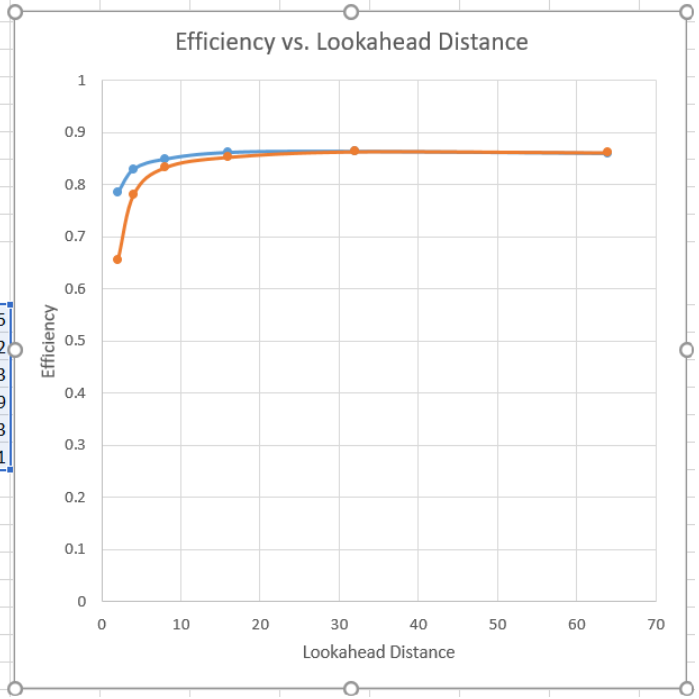
- Stipulate a maximum disparity M, in DATA and in STROBE, that the second-stage encoder will allow
- Second stage will only cycle-stretch as much as necessary to keep the resulting encoded SpaceWire below the maximum disparity allowed.
- This works, it's water-tight (no accumulating disparities over time) and it can be pretty efficient, depending on the input data.

Graphing Disparities over M and over N

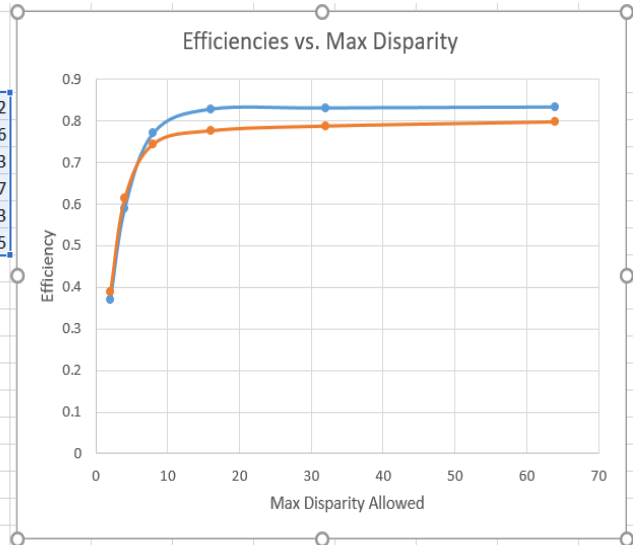
Measured with randomized input data, with no time gaps between packets --

flight data is closer to randomized packets than to a contrived worst case.

LOOKAHEAD DISTANCE	NAÏVE LOOKAHEAD	ROLLING AVERAGE
2	0.78573	0.654965
4	0.830079	0.781572
8	0.849068	0.83363
16	0.861582	0.853439
32	0.863544	0.864023
64	0.859892	0.862111



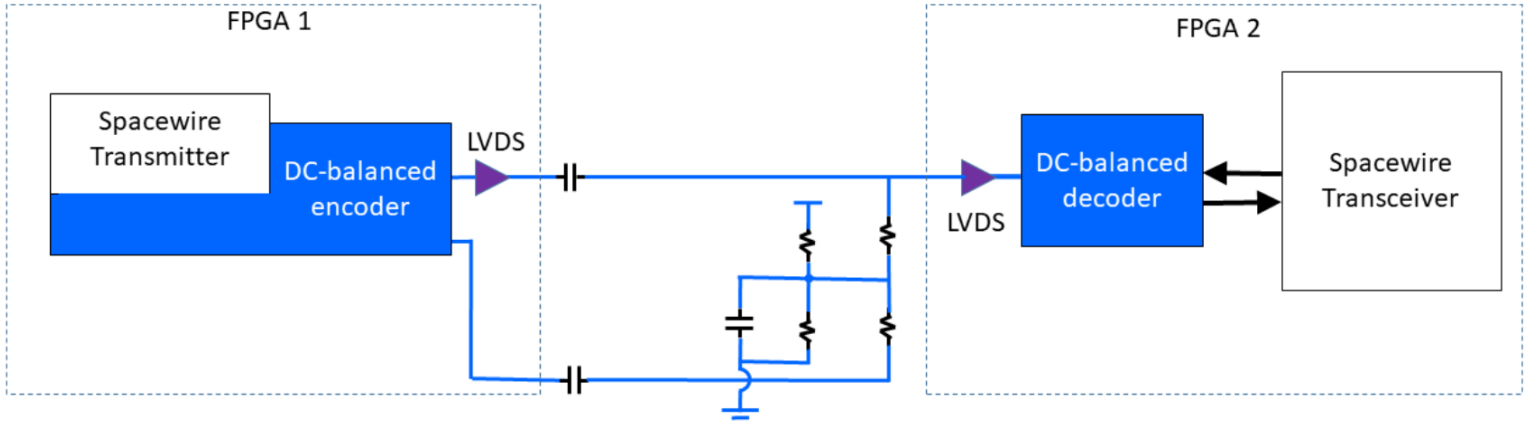
MAX DISPARITY ALLOWED	EFFICIENCY OF NAÏVE LOOKAHEAD METHOD	EFFICIENCY OF ROLLING AVERAGE METHOD
2	0.37087	0.389592
4	0.589992	0.614596
8	0.771133	0.744613
16	0.827826	0.776167
32	0.830571	0.787273
64	0.832938	0.797295



In a practical system, we would use time gaps between packets to zero out any disparities, using NULLs to create opportunities for cycle-stretching.

Follow-on work:

- Full implementation of lookahead encoders incorporated into existing IP.
- Explore range of component values to optimize max disparity and transmission rate.
- Testing against a full range of commercial SpaceWire equipment and a variety of traffic.
- Raise the TRL (technology readiness level) and identify a mission to fly it.
- Consider an encoder that builds its own statistical model of the traffic it sees.
- Ultimately, there is some best cycle-stretch solution for a given M and N; we don't yet know how to find it.



JOHNS HOPKINS
APPLIED PHYSICS LABORATORY

Email: Christopher.Rose@jhuapl.edu